CoDe: a Graphic Language for complex system visualization

P. Ciuccarelli[1], M. I. Sessa[2], M. Tucci[2]

Abstract The concept of *macroscope* has been exploited in several contexts [dR, f] to outline some methodologies aimed to describe the information of complex systems. The basic idea is that complex systems should be looked at as an integrated whole, rather than be described as an aggregate of stand-alone components. Following this approach, we introduce a *Graphic Language* named *CoDe* (*Co*mplexity *De*sign) to describe visual representations which bring information according to the basic idea of a *macroscope*. The syntactic and semantic specification of *CoDe* is a framework for a methodology aimed to support visual representation of complex information. An extensible set of graphical patterns are the lexical elements of the language, which provides two families of operators: a first set of operators is supplied to compose basic patterns to form complex ones; a second set of operators allow the designer to specify the semantics of graphs by creating a logical link with the data they represent. Thus, the *CoDe* Graphic Language is suited to implement the features of such a methodology. *CoDe* is proposed as an interactive tool that support the designer of a visualization project in describing the complex system to be represented, by means of component abstraction and relation discovery, which is the basis for most decision processes. An application to graphic visualization realized by the research group *DensityDesign* at Politecnico di Milano - INDACO Department [dd] is provided. Some future applications to Data Mining and Datawarehouse are also outlined.

1. Introduction

The basic idea introduced by de Rosnay [dR] with the concept of *macroscope* is that a complex system can be better understood if it is described as a whole, rather than as a composition of separate descriptions. As a microscope helps in observing extremely small phenomena, and a telescope make it possible to look at extremely great objects, so a *macroscope* is needed to deal with extremely complex systems.

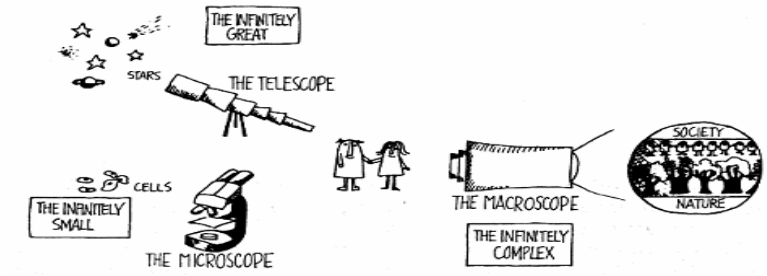


Fig. 1. *The macroscope*

To communicate information, whatever field it belongs to, we always need a proper language. So, also the information that a *macroscope* can extract from a complex system should be expressed in a language suited to describe and to manipulate these kind of data. Usually, complex information is represented using visualization techniques of various kinds, each of them capturing some particular feature and some specific relationship between components. Let us recall that in [B] the concept of *efficiency of a visualization* is discussed and the following definition is proposed: "The most efficient (graphic)

[1] Politecnico di Milano, Dipartimento di Industrial Design, Arti, Comunicazione e Moda (Indaco), Milan, Italy

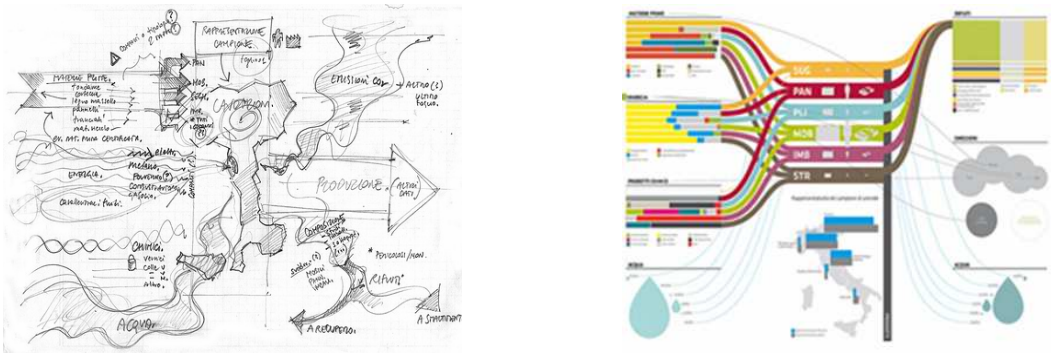
[2] Università di Salerno, Dipartimento di Matematica e Informatica, Salerno, Italy

constructions are those in which any question, whatever its type and level, can be answered in a single instant of perception, that is, in a *single image*." Thus, according to the concept of *macroscope*, we believe that a display can better describe complex information if it can be visualized as a whole by means of a proper graphic language that allows an observer to directly understand the relationships among the considered components, rather than to separately deal with the description of each component.

The *CoDe* (*Co*mplexity *Design*) graphic language we introduce in this paper aims to describe the complex information related to a system by means of a visual approach that follows the idea of *macroscope*. We define the syntactic and semantic specification of *CoDe* to establish a methodological framework to interactively support the observer of the complex information related to a system with proper visual representations. *CoDe* is characterized by an extensible set of graphic patterns that are the lexical elements of the language, that is, the domain of its terms. *CoDe* also provides two families of operators: a first set of operators is supplied to compose basic patterns to form complex ones; a second set of operators allow the designer to specify the semantics of graphs by creating a logical link with the data they represent.

One of the main purposes of the definition of *CoDe* is to support the activities of a designer of the graphic representation of complex systems, by a development environment that facilitates both the process of abstraction of components, and the process of identification of interrelationships, which are the basis for many decision activities. A vast literature has been developed on information visualization. In particular, we consider the contributions of Beertin [B] and Taft [T] for scientific visualization and mapping, and collection of contributions by Card, Mackinlay, and Shneiderman [CMS] in which they focus on using visualization to discover connections and relationships. In addition, a large number of software tools are available to support the processes of design, development and management of various visualization techniques. The definition of a graphical language offers a methodological approach to support the specification and design of the display of complex information, allowing a designer to synthesize unambiguously the architecture as an expression of language. The meta-information representation provided by these expressions can also be used to validate the semantics of the final representation, facilitating the verification of correctness and completeness with respect to the initial specification, and the process of changing and experimenting with different views.

In general, the formal definition of a language requires the identification of a syntax for describing the correct expressions of the language and of a semantics for the attribution of meaning to those expressions. The theory of formal languages is very wide. The objective shared by all approaches is to manage syntactic and semantic information that contains everything is necessary to the description (and possibly to the processing) of information concerning the reality in question. On the other hand, the choice of which approach should be followed in defining a formal language is very critical from the standpoint of implementation.



Project specification *Implementation*

Figure 2. A case study application: graphic design

In the following sections we present the paradigm underlying the syntactic and semantic definition of *CoDe*. We also describe a case study application to a display based on the work carried out by the research team *DensityDesign* at Politecnico di Milano - INDACO Department [dd]. To briefly describe the application field of graphic design, the left side of figure 2 shows a hand-drawn project specification, that is a schematic graphic representation of the relevant information about a given system and of the *meta-information* related to the complexity of interrelationships among the elements that the final implementation has to represent. The implemented graphic representation of the given system appears on

the right side of figure 2. At first visual impact, although there are many similarities between the two views, one can easily detect some relevant differences both in terms of pictorial elements and in terms of links between different components. Using the syntax of *CoDe*, one can translate both views in that language, obtaining a synoptical, unambiguous, representation of the meta-information they contain in terms of abstraction of components and their interrelationships. A comparison of the translations so obtained (in accordance with the syntax and semantics of the language *CoDe*) can help determine whether the final display meets the initial specification in the sense that they are semantically equivalent. In this way *CoDe* becomes the *macroscope-language* shared by the authors of the two visualizations, who use it to view and interpret the same represented information.

2. Syntactic and Semantic Rules of the *CoDe* Graphic Language

The formal definition of *CoDe* is based on the idea that a visual representation of complex systems should be considered as a *statement*. Thus, the formal definition of the language takes into account both syntactic and semantic rules [R-N]. The syntactic part states how *well-formed* visual representations can be constructed, so that the semantic rules can easily provide interpretations based on the related information. Following this approach, the theory of the first order Logic provides a natural framework. Indeed, by the ontological point of view, a graphic visualization can be considered as made by *terms*, that provide information items represented by means of suitable graph types, and *relations* between these terms, that point out relationships and state semantic links with the data they represent. Then, by exploiting terms and relations, the architectural structure of the visualization can be described at a meta-level which provides an abstract representation of the information to represent. On the other hand, the choice of a specific visualization of this information is obtained by selecting a graphical interpretation for the exploited terms and relations. In other words, the same information, in terms of data and relationships, can be represented by very different visualizations, just differing by an aesthetic point of view, depending on the choice of different graph types used to represent information items, or different styles in rendering the relationships. However, if the represented information items and the architectural relationships are the same, by translating these different visualizations in *CoDe*, the equivalence of their meaning, at a suitable level of abstraction, should be made apparent by obtaining the same expression in this graphic meta-language.

In order to define a graphic visualization in *CoDe*, we considered a starting set of *basic graph types*, representing items of information. As an example, if the items of information are the frequencies of some observations, then an Histogram or a Pie can be considered as suitable basic graph types. To increase the information represented by a graph type, we can define other elements in the domain, obtained using composition or transformation *functions* applied to the elements in the basic set. These function are, in some sense, *constructors* of graph types representing more complex items. Both basic and complex graph types are named *terms*, and are the elements of the *domain* of *CoDe*. A semantic interpretation of a term is obtained by stating links to the represented data, according with the intended semantics of the graph types.

In the theory of Logic languages, it is well known the role played by the concept of *Herbrand Universe*. Roughly speaking, this is a set of symbols that can represent the elements of the domain, whatever interpretation is considered. It is very useful to apply this idea also in the context of a language for information visualization. Indeed, the aim is to focus on the abstraction of information items and relations between them provided by the architectural structure of the visualization, and not on the specific shapes selected for the corresponding graph types or relationship links. Thus, we introduce graphical representation symbols that, in some sense, play the role of a Herbrand Universe in the context of *CoDe*. These symbols allows us to denote the meaning carried out by a visualization, without specifying the shape of a particular graph type or link chosen to represent this information. As an example, if we consider a distribution of frequency values as a information item, a Histogram is a suitable choice of graph type for the related visualization. The shape of the Histogram can depend by aesthetic factors, but the meaning is always the same. Then, the symbol exploited in *CoDe* to denote the histogram provides only information related to the structural features represented by this graph type, apart from his shape in the final instantiation.

Visual Notation for Terms

A term represents an item of information related to some objective data. As a *starting set of terms*, we consider the *set of standard-graphs* (Histogram, Pie, Area, Bubble, Line, Radar), that are widely used to visualize a *double-entry table*:

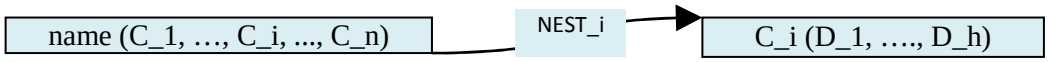
Title_name		
C_1	.....	C_n
Value_1	.....	Value_n

Let us recall that any standard-graph has a fixed structural rule that states, by a proportional magnitude, the semantic correspondence between the value *Value_i* of a component *C_i* and its visualization on the plane. On the other hand, the final implementation of a standard-graph depends on many visual parameters like texture, color, orientation, shape, ecc… A term of *CoDe* only concerns the structural features of the considered information. In the sequel, a term is denoted by a blue rectangle, containing a label that shows its name. In the case of a term representing a standard-graph, the related components are given in parenteses:

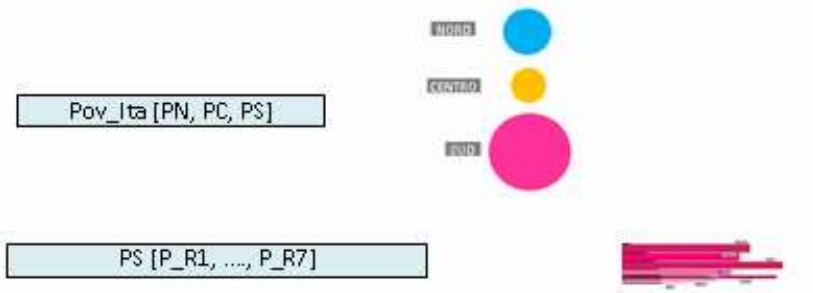
name_data (C1, …Ck)

Several *functions* allow a designer to construct more complex terms, exploiting composition and modification of available terms, that can also overlap by sharing part of the representation area. The symbol used in *CoDe* to denote a function is a directed line, which connects the involved terms. A blue label provides the name of the applied function. In the sequel, we give some examples of functions. Examples of visualizations of the obtained more complex terms, based on *DensityDesign* graphs, are also given.

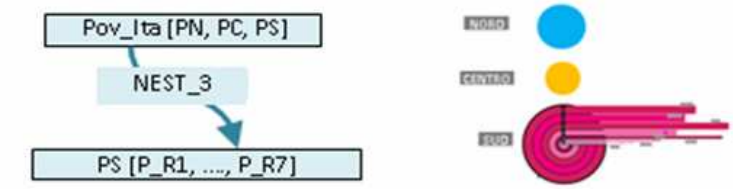
As an example of operator that can be defined to construct more complex terms, let us show the function *NEST_i* on graph types, where the value represented by the *i-th* component *C_i* of one type are further specified by the other one at a deeper level of detail. The visualization of the more complex term of *CoDe* is as follows:



The terms involved by the *NEST_i* function can be visualized by the same or by different standard-graphs. For instance, let us consider as given terms *Pov_Ita (PN, PC, PS)* and *PS (P_R1, …, P_R7)*, which provide, separately, distribution of poor people in the North, Center and South parts of Italy, and distribution of poor people in the 7 regions of South Italy. Below, the representation of the corresponding terms in *CoDe* language are shown, together with the related visualizations by means of standard-graphs Bobble and Histogram, respectively:



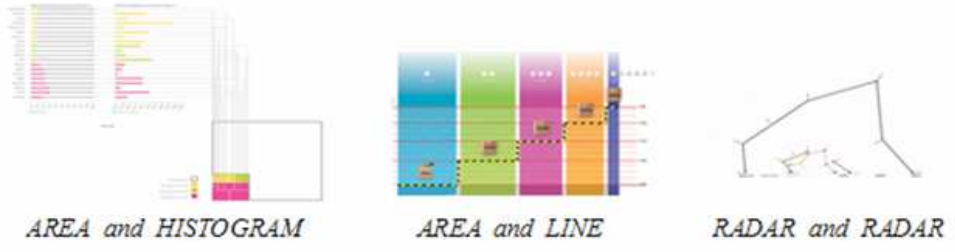
By applying the *NEST_3* function, we can construct a more complex information item which organizes, in a single structure, the information given by the two separate items. The semantics of the new term, which represents this more complex information item, can be constructed and visualized in *CoDe* language according to the definition of the *NEST_i* function. In the considered example, the visualization of the more complex term can be obtained by suitably overlapping the two standard graphs:



It is also possible to choose the same standard graph to visualize the given terms. Choosing Histogram standard-graphs for both, the following visualizations, respectively, of the given terms and of the more

complex term, can be obtained. In this case, the two standard graphs are composed in sequence, without overlapping (the regions are 8 because Sardinia is also considered):

To see the capabilities of this kind of function, we show some other examples of application of the *NEST_i* function, always from *DensityDesign* visualizations, providing the related standard-graphs that are involved:



Another example of operator is the *aggregation* function, which involves terms that have the same components. They are considered as a single term, but preserving their distinct identities. In *CoDe*, a term obtained by applying the aggregation function is denoted by a dashed blue label including both the involved terms and the *AGGR name* symbol, which indicate the applied function and a name related to the intended semantics of the grouping. Moreover, when the description of the meta-information is made at an higher abstraction level, *AGGR_name* term can be represented without specifying the involved terms, as in the following example:

Finally, we introduce a different kind of function, named *visualization operator*, which only involves one term and does not concern its structure. Indeed, it only modifies parameters of the visualization of the involved term, to improve the efficiency and the aesthetic impact of visual representation. For example, the *i-th* component of a distribution of frequency values could be visualized by a *metaphoric icon*, or a *metaphoric color intensity* that provides a proportional visualization of the related value. Because of the different roles played by these interface operators, they have different notations. The name of the applied operator is denoted with a green label attached to the term, and the involved components are given in parentheses.

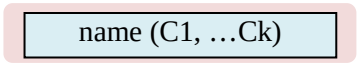
As an example of implementation, in the visualization of the previous term *PS [P_R1, …, P_R8]*, instead of using names of Italian regions to denote components, the related geographic shapes can be considered. On the other hand, if numeric values providing measures of water resource are associated with the components, drops of proportional size can be used to visualize the proportion between these values, as follows:

Visual Notation for Relations

In *CoDe* two kinds of relations are considered: the *link relation*, that allows to state the existence of a logical relationship between terms, and the *set relation*, that allows to assert the information expressed by

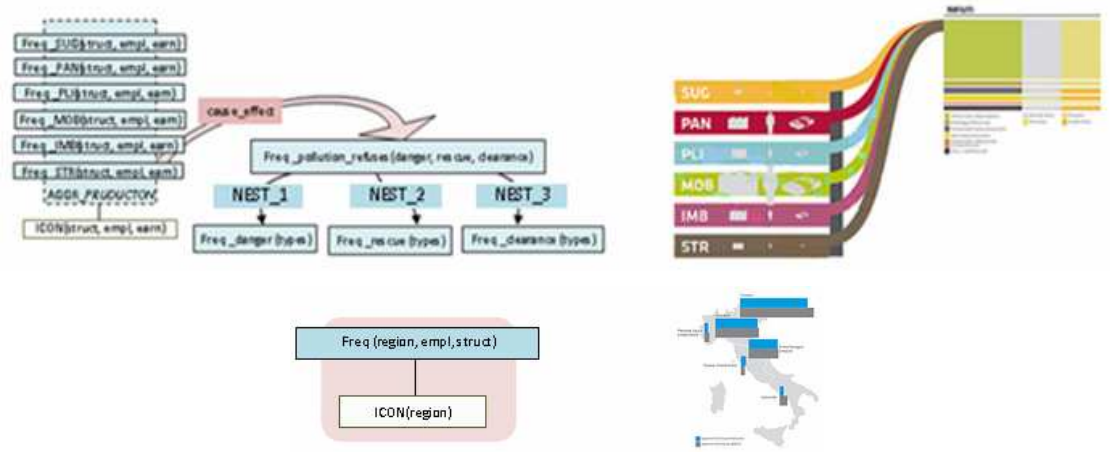
a term. The symbol used by *CoDe* to denote a *link* is a directed red arrow, which connects the component terms. A red label denotes the intended meaning of the relation:

For the monadic relation *set*, the symbol is a red smoothed rectangle containing the asserted term:

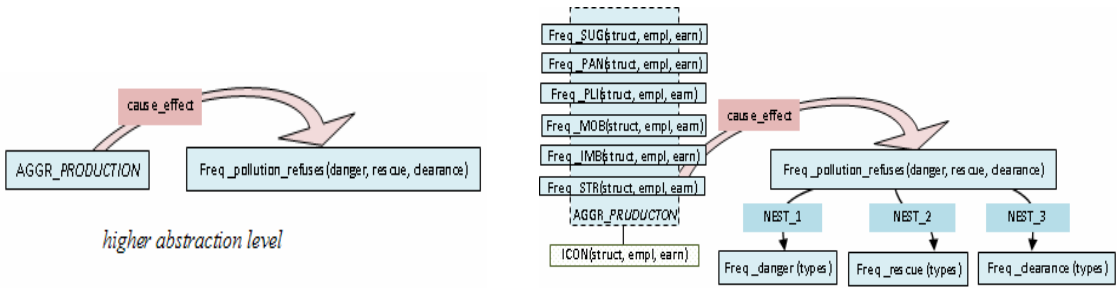


It is useful to stress the difference between functions and relations. A function is a fixed construction tool to carry out a new term exploiting the involved terms. On the other hand, a relation is a predicative tool to express the existence or not of a fixed link between terms, then it provides a truth value, not a new term. Thus, a term is the abstraction of an "objective" data belonging to the complex system that must be visualized, whereas a relation is the abstraction of a logical relationship on data (true of false). Visualized relations are generally intended as stating *true* statement.

To state a relation between terms there are no constraints on their structure, and the chosen relations depend on the "subjective view" of the designer. In other words, the same data can be involved in different relations to emphasize different informative needs. As a consequence, if the designer makes a wrong choice of the relationships to be represented, the information carried out by the related visualization is useless or confusing. The ability to establish relationships is very useful early in the process of visualization, i.e., at the highest level of abstraction, when key components of complex systems are constructed and organized in a coherent and efficient architecture. In subsequent steps, these components can be refined until the ground visualization is reached (the level of data). Examples of *link* and *set* relations represented in *CoDe* are shown below, and possible corresponding visualizations are also provided:



Finally, let us stress that the *CoDe* language allows the description of the architectural structure of information visualization at different levels of abstraction. In the following example, the description at a higher abstraction level on the left, can provide the final visualization on the right through several refinement levels:



Statements of the CoDe Language

Let us recall that a *statement* of a logical language is obtained by composing the truth values of relations according to some operators, named *connectives*, that carry out truth values as results. In *CoDe*, a statement is obtained by representing relations in the same visualization. This statement has the intended semantics of *conjunction of the truth values* of the relations (the AND logic connective). At a given meta-level of abstraction, this graphic statement describes the architecture of the visualization and the related meta-information, both by the syntactic and semantic point of view. However, as stressed in [B], there are human physical bounds in understanding an *image*, considered as "the minimum meaningful visual form perceptible in the minimum instant of vision", and the growth of signs and relations decreases the efficiency of the visualization. Thus, *CoDe* should be used as a tool to organize the architectural structure of visualization efficiently. On the other hand, since *CoDe* is a formal coding system for the visualized meta-information, then a computerized processing, that allows to overcome human limits in managing complexity, is possible. As a tool supporting the visualization process, a representation in *CoDe* allows to manage parts of the graphical visualizations as distinct items. Then, it can enhance the creative experimentation of the designer, by making modifications of the architectural structure or the visualization parameters much simpler. Moreover, some computerized semantic analysis of the visualizations is also possible. Indeed, we can store graphical representations of data both in a format of images, and in formal syntactic of statements in *CoDe*, that are more suitable inputs for computer algorithms of information retrieval or data mining. With this aims, let us state the following Definitions useful to compare visualizations as shown in the next section.

Definition 1*:* Two visualizations of the same complex system are *congruent* if there exists an abstraction level such that the related translations in *CoDe* language have the same terms and relations.

Definition 2*:* Two visualizations of the same complex system are *equivalent* if the related translations in *CoDe* language, at the ground data level, are congruent.

3. Application to Graphic Design

In this section a translation in *expressions* of *CoDe* language for both visualizations in Figure 2 is provided.
According with *Definition 1*, it can be verified that the translations in Figure 3 and Figure 4, are *congruent* (i.e. contain the same terms and relations). Thus, at this level of abstraction, the final implementation is a visualization correct and complete w.r.t. the required specification of information.

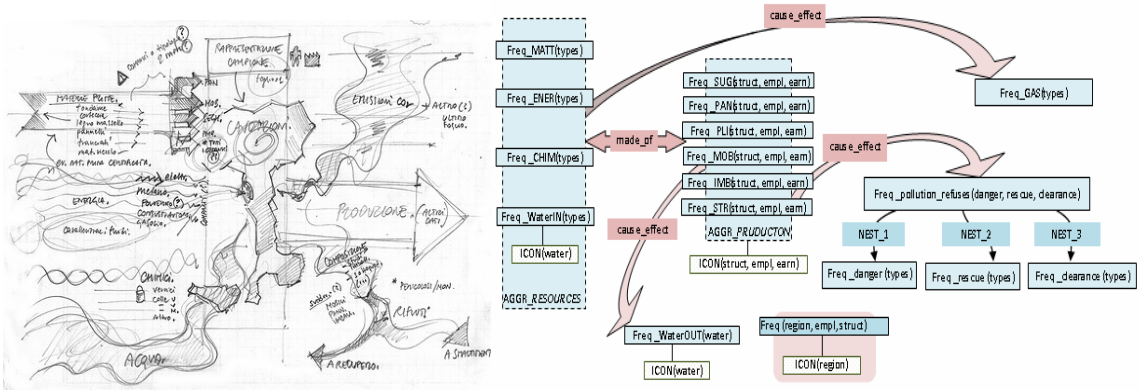


Fig. 3. A) Project specification B) Translation in *CoDe* language

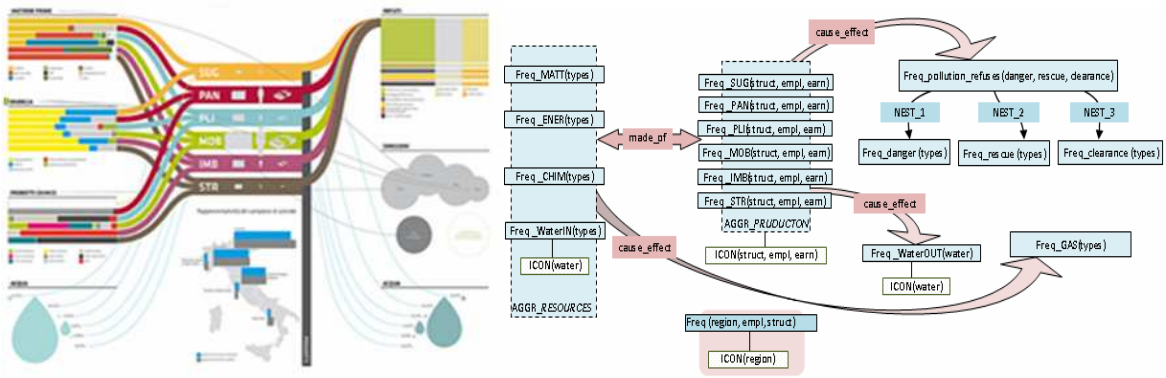


Fig. 4. A) Project implementation B) Translation in *CoDe* language

According with the concept of *efficiency of a visualization* given in section 1, Figures 3 and 4 allow us to stress this concept at the level of meta-information carried out by the translation in *CoDe* language. Moreover, they show the contribution of the visualization choices, made by the designers, to improve the efficiency of the visualization at the ground level of the considered data.

4. Concluding Remarks and Further Work

Taking the concepts set out in the introduction, we might note that one of the purposes of using a graphical language would be to serve as a *macroscope language* providing an encoding of the meta-information related to the architecture of a display supporting decision processes. The term *architecture* is commonly used not only in relation to the structure of buildings and urban organization, but also in relation to the structure of other complex systems, as in the case of computer hardware. In this context, it is well known by a designer of hardware the use a graphic languages to display the hardware components and their interconnections, especially in the software interactive tools of CAD (computer aided design).

Remaining in the field of decision support systems, a research activity on the use of the graphic language *CoDe* in the field of Data Mining and Datawarehousing is in progress. In fact, according to the characteristics of the graphical language described in the previous sections, it is possible to connect a graph module to the cubes of a datawarehouse and perhaps we could consider the operators as visual function for cube composition. Therefore, we aim at using CoDe as a tool to describe the structure of the graphic representation of information extracted from a datawarehouse. It should be emphasized that, in these terms, we only get a technical view of information, not definable as a Data Mining or Machine Learning technique. However, considering the language in an interactive environment, we can exploit the displaying of modules and operators as a support both for the abstraction process leading to the identification of relevant components of a complex system and for the process of identifying relationships between components. Both processes underly many decision activities conducted by humans.

Bibliography

[dR] Joël de Rosnay, *Le Macroscope. Vers une vision globale*, Editions du Seuil, 1975, (English translation: *The Macroscope: A New World Scientific System*, Harper & Row Publishers, New York, 1979).

[T] Edward R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire CT, 1983.

[B] Jacques Bertin, *Sémiologie Graphique. Les diagrammes, les réseaux, les cartes*, with Marc Barbut [et al.], Gauthier-Villars, Paris, 1967, (English translation: *Semiology of Graphics: Diagrams, Networks, Maps*, 1983) .

[CMS] Stuart K. Card, Jock Mackinlay, Ben Shneiderman (Editors), *Readings in Information Visualization: Using Vision to Think*, Academic Press, San Diego CA, 1999.

[RN] Stuart Russel, Peter Norvig, *Artificial Intelligence: a modern approach*, third ed., Prentice Hall, Upper Saddle River, NJ, 2009.

[dd] http://www.densitydesign.org